

## 日 本 国 特 許 庁

PATENT OFFICE  
JAPANESE GOVERNMENTJ1017 U.S. PTO  
09/803168  
03/09/01

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

2000年 3月 9日

出 願 番 号

Application Number:

特願2000-064718

出 願 人

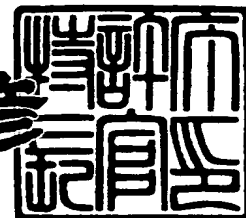
Applicant(s):

インターナショナル・ビジネス・マシーンズ・コーポレーション

2000年 6月29日

特許庁長官  
Commissioner,  
Patent Office

近 藤 隆 彦



出証番号 出証特2000-3051316

【書類名】 特許願

【整理番号】 JA999286

【提出日】 平成12年 3月 9日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 12/00

【発明者】

【住所又は居所】 神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ピー・エム株式会社 東京基礎研究所内

【氏名】 河内谷 清久仁

【発明者】

【住所又は居所】 神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ピー・エム株式会社 東京基礎研究所内

【氏名】 小野寺 民也

【特許出願人】

【識別番号】 390009531

【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】 100086243

【弁理士】

【氏名又は名称】 坂口 博

【復代理人】

【識別番号】 100104880

【弁理士】

【氏名又は名称】 古部 次郎

【選任した代理人】

【識別番号】 100091568

【弁理士】

【氏名又は名称】 市位 嘉宏

【選任した復代理人】

【識別番号】 100100077

【弁理士】

【氏名又は名称】 大場 充

【手数料の表示】

【予納台帳番号】 081504

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9304391

【包括委任状番号】 9304392

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 コンピュータシステム、メモリ管理方法、記憶媒体及びプログラム伝送装置

【特許請求の範囲】

【請求項 1】 プログラムを複数の実行主体に分けて実行すると共に、複数の当該実行主体がメモリ装置に格納されているデータを共有してそれぞれアクセスするようなデータ処理環境を有するコンピュータシステムにおいて、

前記メモリ装置に格納されている前記データは、単一の前記実行主体のみから直接アクセスされる状態である局所性を有するか否かを示すフラグデータを有し

、  
前記実行主体は、アクセスしようとする前記データにおける前記フラグデータが、当該実行主体に対して局所性を有することを示す場合は、他の実行主体からのアクセスを拒否するロック処理を行わずに当該データにアクセスし、当該フラグデータが、当該実行主体に対して局所性を有さないことを示す場合は、当該ロック処理を行った後に当該データにアクセスすることを特徴とするコンピュータシステム。

【請求項 2】 前記実行主体は、前記メモリ装置に格納されている前記データの中から、前記フラグデータが当該実行主体に対して局所性を有することを示しているデータであって、当該実行主体が参照ポインタを持たないデータを検出し、自由に使用できる記憶領域として解放することを特徴とする請求項 1 に記載のコンピュータシステム。

【請求項 3】 複数のスレッドがオブジェクトを共有してそれぞれアクセスするようなデータ処理環境を有するコンピュータシステムにおいて、

前記オブジェクトは、単一の前記スレッドのみから直接アクセスされる状態である局所性を有するか否かを示すフラグデータを有し、

前記スレッドは、アクセスしようとする前記オブジェクトにおける前記フラグデータが、当該スレッドに対して局所性を有することを示す場合は、他のスレッドまたは前記オブジェクトからのアクセスを拒否するロック処理を行わずに当該オブジェクトにアクセスし、当該フラグデータが、当該スレッドに対して局所性

を有さないことを示す場合は、当該ロック処理を行った後に当該オブジェクトにアクセスすることを特徴とするコンピュータシステム。

【請求項4】 前記オブジェクトは、生成された際に、当該オブジェクトを生成した前記スレッドに対して局所性を有することを示すように前記フラグデータをセットし、他のスレッドまたはオブジェクトからアクセス可能な状態に変化させられる際に、その状態変化に先立って前記フラグデータの示す局所性を解除することを特徴とする請求項3に記載のコンピュータシステム。

【請求項5】 前記スレッドは、前記フラグデータが当該スレッドに対して局所性を有することを示している前記オブジェクトであって、当該スレッドが参照ポインタを持たないオブジェクトを検出し、メモリ装置上における自由に使用できる記憶領域として解放することを特徴とする請求項3に記載のコンピュータシステム。

【請求項6】 プログラムを複数の実行主体に分けて実行すると共に、複数の当該実行主体がオブジェクトを共有してそれぞれアクセスするようなデータ処理環境におけるメモリ管理方法であって、

前記オブジェクトを生成する際に、当該オブジェクトを生成した前記実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットするステップと、

所定の前記実行主体に対して局所性を有する前記オブジェクトが、当該実行主体によって他の実行主体からアクセス可能な状態に変化させられる際に、その状態変化に先立って前記フラグデータにて示される局所性を解除するステップと、

前記実行主体が所定の前記オブジェクトにアクセスしようとする際に、当該オブジェクトの前記フラグデータが、当該実行主体に対して局所性を有することを示す場合は、他の実行主体またはオブジェクトからのアクセスを拒否するロック処理を行わずに当該オブジェクトにアクセスし、当該フラグデータが、当該実行主体に対して局所性を有さないことを示す場合は、当該ロック処理を行った後に当該オブジェクトにアクセスするステップとを含むことを特徴とするメモリ管理方法。

【請求項7】 前記オブジェクトの前記フラグデータにて示される局所性を

解除するステップは、

前記オブジェクトが所定の前記実行主体に対して局所性を有していた際に当該実行主体からのアクセスにおいて省略されたロック処理を実行するステップを含むことを特徴とする請求項6に記載のメモリ管理方法。

【請求項8】 プログラムを複数の実行主体に分けて実行すると共に、複数の当該実行主体がオブジェクトを共有してそれぞれアクセスするようなデータ処理環境におけるメモリ管理方法であって、

前記オブジェクトを生成する際に、当該オブジェクトを生成した前記実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットするステップと、

前記実行主体が、前記フラグデータが当該実行主体に対して局所性を有することを示している前記オブジェクトであって、当該実行主体が参照ポインタを持たないオブジェクトを検出するステップと、

検出されたオブジェクトをメモリ装置上における自由に使用できる記憶領域として解放するステップとを含むことを特徴とするメモリ管理方法。

【請求項9】 コンピュータに実行させるプログラムを当該コンピュータの入力手段が読取可能に記憶した記憶媒体において、

前記プログラムは、

オブジェクトを生成する際に、当該オブジェクトを生成した前記実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットする処理と、

所定の前記実行主体に対して局所性を有する前記オブジェクトが、当該実行主体によって他の実行主体からアクセス可能な状態に変化させられる際に、その状態変化に先立って前記フラグデータにて示される局所性を解除する処理と、

前記実行主体が所定の前記オブジェクトにアクセスしようとする際に、当該オブジェクトの前記フラグデータが、当該実行主体に対して局所性を有することを示す場合は、他の実行主体またはオブジェクトからのアクセスを拒否するロック処理を行わずに当該オブジェクトにアクセスし、当該フラグデータが、当該スレッドに対して局所性を有さないことを示す場合は、当該ロック処理を行った後に

当該オブジェクトにアクセスする処理とを前記コンピュータに実行させることを特徴とする記憶媒体。

【請求項 1 0】 コンピュータに実行させるプログラムを当該コンピュータの入力手段が読取可能に記憶した記憶媒体において、

前記プログラムは、

オブジェクトを生成する際に、当該オブジェクトを生成した前記実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットする処理と、

前記実行主体が、前記フラグデータが当該実行主体に対して局所性を有することを示している前記オブジェクトであって、当該実行主体が参照ポインタを持たないオブジェクトを検出する処理と、

検出されたオブジェクトをメモリ装置上における自由に使用できる記憶領域として解放する処理とを前記コンピュータに実行させることを特徴とする記憶媒体

。

【請求項 1 1】 コンピュータに、

オブジェクトを生成する際に、当該オブジェクトを生成した前記実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットする処理と、所定の前記実行主体に対して局所性を有する前記オブジェクトが、当該実行主体によって他の実行主体からアクセス可能な状態に変化させられる際に、その状態変化に先立って前記フラグデータにて示される局所性を解除する処理と、前記実行主体が所定の前記オブジェクトにアクセスしようとする際に、当該オブジェクトの前記フラグデータが、当該実行主体に対して局所性を有することを示す場合は、他の実行主体またはオブジェクトからのアクセスを拒否するロック処理を行わずに当該オブジェクトにアクセスし、当該フラグデータが、当該スレッドに対して局所性を有さないことを示す場合は、当該ロック処理を行った後に当該オブジェクトにアクセスする処理とを実行させるプログラムを記憶する記憶手段と、

前記記憶手段から前記プログラムを読み出して当該プログラムを送信する送信手段とを備えたことを特徴とするプログラム伝送装置。

【請求項 1 2】 コンピュータに、

オブジェクトを生成する際に、当該オブジェクトを生成した前記実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットする処理と、前記実行主体が、前記フラグデータが当該実行主体に対して局所性を有することを示している前記オブジェクトであって、当該実行主体が参照ポイントを持たないオブジェクトを検出する処理と、検出されたオブジェクトをメモリ装置上における自由に使用できる記憶領域として解放する処理とを実行させるプログラムを記憶する記憶手段と、

前記記憶手段から前記プログラムを読み出して当該プログラムを送信する送信手段とを備えたことを特徴とするプログラム伝送装置。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、プログラムを複数の実行主体（スレッド）に分けて実行すると共に、当該複数の実行主体がそれぞれ共有されたデータにアクセスするようなデータ処理環境を有するコンピュータシステムに関する。

【0 0 0 2】

【従来の技術】

コンピュータシステムにおいて、プログラムを複数の実行主体（スレッド）に分けて実行すると共に、当該複数の実行主体がそれぞれ共有されたデータにアクセスするようなデータ処理環境を考える。この種の処理環境では、所定の実行主体が所定のデータにアクセスする場合、当該データを他の実行主体が使用しないようにするロック（同期）処理を行う。このため、反対に、アクセスしようとするデータが他の実行主体によりアクセスされ、ロック（同期）されている場合は、アンロック（解放）されるまで待つこととなる。

【0 0 0 3】

このような処理環境では、特定の単一の実行主体以外からはアクセスされないデータが存在する場合がある。しかし、そのようなデータへのアクセスにおいても、当該実行主体がアクセスする場合にはロック（同期）処理が行われていた。



すなわち、他の実行主体からアクセス要求される可能性のない場合にまで、ロック（同期）処理を行うという無駄な動作が行われていた。

【0004】

また、この種の処理環境では、データを格納したメモリ装置の記憶領域において、いずれの実行主体からもアクセスされなくなったデータが残ってしまい、記憶領域が不必要に減少してしまう場合がある。そこで、必要に応じて、メモリ装置の記憶領域からそのような使用されなくなったデータをまとめて除去し、使用可能な記憶領域を増やすガーベジ・コレクションを行っている。

従来、ガーベジ・コレクションの手法としては、実行主体によるプログラムの実行を一度全て停止させ、記憶領域にある全てのデータの中からいずれの実行主体からもアクセスされなくなったデータを検索し除去する方法や、実行主体によるプログラムの実行と並行して少しずつ各データへのアクセスの有無を調べ、いずれの実行主体からもアクセスされなくなったデータを検索し除去する方法が採られている。

【0005】

この種のデータ処理環境の例として、Java言語で記述されたプログラムの動作環境（以下、Javaの動作環境と称す）がある。Java言語では、プログラムの実行主体であるスレッドが、メモリ装置の記憶領域に設定されたオブジェクトプール上に生成されるオブジェクトにアクセスすることにより、全ての処理が実行される。したがって、Javaの動作環境では、ロック（同期）処理やガーベジ・コレクションが頻繁に発生する。

そのため、ロック（同期）処理を高速化する技術や、ガーベジ・コレクションに要する時間を短縮する技術の開発が望まれている。

【0006】

【発明が解決しようとする課題】

上述したように、複数の実行主体（スレッド）が共有されたデータ（オブジェクト）にアクセスするようなデータ処理環境では、特定の単一の実行主体からのみアクセスが行われ、他の実行主体からアクセス要求される可能性のないデータに対しても、アクセスする際にロック（同期）処理を行うという無駄な動作が行

われていたため、システムの全体性能を低下させていた。

【0007】

また、ガーベジ・コレクションにおいては、実行主体によるプログラムの実行を一度全て停止させて不要なデータを除去する場合は、当該処理を行う際にシステム上の他の一切の処理を中断しなくてはならないため、ユーザにとって動作が煩雑となっていた。

【0008】

一方、実行主体によるプログラムの実行と並行して不要なデータを除去する処理を実行する場合は、通常の処理を実行しながら、合間にデータの検索及び除去を実行するため、システムの全体性能を低下させてしまう。

【0009】

そこで本発明は、単一の実行主体からのみアクセスが行われ、他の実行主体からアクセス要求される可能性のないデータに対しては、アクセスする際にロック（同期）処理を省略することにより、システムの負荷を軽減し、全体的な性能を向上させることを目的とする。

【0010】

また本発明は、通常の処理と並行して行うガーベジ・コレクションの効率化を図ることにより、システムの全体性能の低下を軽減することを他の目的とする。

【0011】

【課題を解決するための手段】

かかる目的のもと、本発明は、プログラムを複数の実行主体に分けて実行すると共に、この複数の実行主体がメモリ装置に格納されているデータを共有してそれぞれアクセスするようなデータ処理環境を有するコンピュータシステムにおいて、メモリ装置に格納されているデータは、単一の実行主体のみから直接アクセスされる状態である局所性を有するか否かを示すフラグデータを有し、実行主体は、アクセスしようとするデータにおけるこのフラグデータが、この実行主体に対して局所性を有することを示す場合は、他の実行主体からのアクセスを拒否するロック処理を行わずにこのデータにアクセスし、このフラグデータが、この実行主体に対して局所性を有さないことを示す場合は、ロック処理を行った後にこ

のデータにアクセスすることを特徴としている。

自実行主体に対して局所性を有するデータにアクセスする際に、ロック処理を省略することにより、実行主体の負荷を軽減し、システム全体の性能を向上させることが可能となる。

【0012】

ここで、この実行主体は、メモリ装置に格納されているデータの中から、フラグデータがこの実行主体に対して局所性を有することを示しているデータであって、この実行主体が参照ポインタを持たないデータを検出し、自由に使用できる記憶領域として解放することを特徴としている。

このような構成とすれば、実行主体から参照されないデータを消去して記憶領域を解放するための処理を、所定の実行主体においてローカルに実行できるため、他の実行主体による通常の処理を停止することなく、並行して行うことができる点で好ましい。

【0013】

また、本発明は、複数のスレッドがオブジェクトを共有してそれぞれアクセスするようなデータ処理環境を有するコンピュータシステムにおいて、オブジェクトは、単一のスレッドのみから直接アクセスされる状態である局所性を有するかどうかを示すフラグデータを有し、スレッドは、アクセスしようとするオブジェクトにおけるフラグデータが、このスレッドに対して局所性を有することを示す場合は、他のスレッドまたはオブジェクトからのアクセスを拒否するロック処理を行わずにこのオブジェクトにアクセスし、このフラグデータが、このスレッドに対して局所性を有さないことを示す場合は、ロック処理を行った後にこのオブジェクトにアクセスすることを特徴としている。

【0014】

ここで、このオブジェクトは、生成された際に、このオブジェクトを生成したスレッドに対して局所性を有することを示すようにフラグデータをセットし、他のスレッドまたはオブジェクトからアクセス可能な状態に変化させられる際に、その状態変化に先立って前記フラグデータにて示される局所性を解除することを特徴としている。すなわち、全てのオブジェクトは初期的にはいずれかのスレッ

ドに対して局所性を有し、所定の状況が発生した場合に、当該スレッドによるアクセスに関する状態変化に応じてこの局所性を解除する。なお、オブジェクトの状態を変化させるべき状況が発生したことをスレッドが検出する方法としては、例えばこのオブジェクトが他のオブジェクトもしくはグローバルなデータに代入された場合とすることができる。さらにこのとき、代入先のオブジェクトが局所的かどうかまで総括的にチェックして解除するようにしても良い。

## 【0015】

さらにこのスレッドは、フラグデータがこのスレッドに対して局所性を有することを示しているオブジェクトであって、このスレッドが参照ポインタを持たないオブジェクトを検出し、メモリ装置上における自由に使用できる記憶領域として解放することを特徴としている。

## 【0016】

また、本発明は、プログラムを複数の実行主体に分けて実行すると共に、この複数の実行主体がオブジェクトを共有してそれぞれアクセスするようなデータ処理環境におけるメモリ管理方法であって、オブジェクトを生成する際に、このオブジェクトを生成した実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットするステップと、所定の前記実行主体に対して局所性を有する前記オブジェクトが、当該実行主体によって他の実行主体からアクセス可能な状態に変化させられる際に、その状態変化に先立って前記フラグデータにて示される局所性を解除するステップと、実行主体が所定のオブジェクトにアクセスしようとする際に、このオブジェクトのフラグデータが、この実行主体に対して局所性を有することを示す場合は、他の実行主体またはオブジェクトからのアクセスを拒否するロック処理を行わずにこのオブジェクトにアクセスし、このフラグデータが、この実行主体に対して局所性を有さないことを示す場合は、このロック処理を行った後に当該オブジェクトにアクセスするステップとを含むことを特徴としている。

自実行主体に対して局所性を有するオブジェクトにアクセスする際に、ロック処理を省略することにより、システム全体の処理速度を高速化させることが可能となる。

## 【 0 0 1 7 】

ここで、このオブジェクトのフラグデータにて示される局所性を解除するステップは、このオブジェクトが所定の実行主体に対して局所性を有していた際にこの実行主体からのアクセスにおいて省略されたロック処理を実行するステップを含むことを特徴とする。

このような構成とすれば、所定のオブジェクトにおいて局所性が解除された場合に、ロック処理を省略したことによって処理の整合性が崩れることを回避することができる点でより優れている。

省略されたロック処理を実行するためには、このロック処理が省略されたことを認識することが必要であるが、これは、実行主体の実行スタックを参照したり、ロック処理を省略するたびに省略した回数をカウントしておくことにより可能である。

## 【 0 0 1 8 】

さらにまた、本発明は、プログラムを複数の実行主体に分けて実行すると共に、この複数の実行主体がオブジェクトを共有してそれぞれアクセスするようなデータ処理環境におけるメモリ管理方法であって、オブジェクトを生成する際に、このオブジェクトを生成した実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットするステップと、実行主体が、このフラグデータがこの実行主体に対して局所性を有することを示しているオブジェクトであって、この実行主体が参照ポインタを持たないオブジェクトを検出するステップと、検出されたオブジェクトをメモリ装置上における自由に使用できる記憶領域として解放するステップとを含むことを特徴としている。

このような構成とすれば、ガーベジ・コレクションを、所定の実行主体においてローカルに実行できるため、他の実行主体による通常の処理を停止することなく、並行して行うことができる点で好ましい。

## 【 0 0 1 9 】

また、本発明は、コンピュータに実行させるプログラムをこのコンピュータの入力手段が読取可能に記憶した記憶媒体において、このプログラムは、オブジェクトを生成する際に、このオブジェクトを生成した実行主体のみからアクセスさ

れる状態である局所性を有することを示すフラグデータをセットする処理と、所定の前記実行主体に対して局所性を有する前記オブジェクトが、当該実行主体によって他の実行主体からアクセス可能な状態に変化させられる際に、その状態変化に先立って前記フラグデータにて示される局所性を解除する処理と、実行主体が所定のオブジェクトにアクセスしようとする際に、このオブジェクトのフラグデータが、この実行主体に対して局所性を有することを示す場合は、他の実行主体またはオブジェクトからのアクセスを拒否するロック処理を行わずにこのオブジェクトにアクセスし、このフラグデータが、このスレッドに対して局所性を有さないことを示す場合は、ロック処理を行った後にこのオブジェクトにアクセスする処理とをこのコンピュータに実行させることを特徴としている。

このような構成とすれば、このプログラムをインストールした全てのコンピュータにおいて、自実行主体に対して局所性を有するオブジェクトにアクセスする際に、ロック処理を省略することにより、システム全体の処理速度を高速化させることが可能となる。

#### 【0020】

コンピュータに実行させるプログラムをこのコンピュータの入力手段が読取可能に記憶した記憶媒体において、このプログラムは、オブジェクトを生成する際に、このオブジェクトを生成した実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットする処理と、実行主体が、このフラグデータがこの実行主体に対して局所性を有することを示しているオブジェクトであって、この実行主体が参照ポインタを持たないオブジェクトを検出する処理と、検出されたオブジェクトをメモリ装置上における自由に使用できる記憶領域として解放する処理とを前記コンピュータに実行させることを特徴している。

このような構成とすれば、このプログラムをインストールした全てのコンピュータにおいて、ガーベジ・コレクションを、所定の実行主体においてローカルに実行できるため、他の実行主体による通常の処理を停止することなく、並行して行うことが可能となる。

#### 【0021】

さらに、本発明は、コンピュータに、オブジェクトを生成する際に、このオブ

ジェクトを生成した実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットする処理と、所定の前記実行主体に対して局所性を有する前記オブジェクトが、当該実行主体によって他の実行主体からアクセス可能な状態に変化させられる際に、その状態変化に先立って前記フラグデータにて示される局所性を解除する処理と、実行主体が所定のオブジェクトにアクセスしようとする際に、このオブジェクトのフラグデータが、この実行主体に対して局所性を有することを示す場合は、他の実行主体またはオブジェクトからのアクセスを拒否するロック処理を行わずにこのオブジェクトにアクセスし、このフラグデータが、このスレッドに対して局所性を有さないことを示す場合は、ロック処理を行った後にこのオブジェクトにアクセスする処理とを実行させるプログラムを記憶する記憶手段と、この記憶手段からこのプログラムを読み出してこのプログラムを送信する送信手段とを備えたことを特徴としている。

このような構成とすれば、このプログラムをダウンロードして実行する全てのコンピュータにおいて、自実行主体に対して局所性を有するオブジェクトにアクセスする際に、ロック処理を省略することにより、システム全体の処理速度を高速化させることが可能となる。

#### 【0022】

さらにまた、本発明は、コンピュータに、オブジェクトを生成する際に、このオブジェクトを生成した実行主体のみからアクセスされる状態である局所性を有することを示すフラグデータをセットする処理と、実行主体が、このフラグデータがこの実行主体に対して局所性を有することを示しているオブジェクトであって、この実行主体が参照ポインタを持たないオブジェクトを検出する処理と、検出されたオブジェクトをメモリ装置上における自由に使用できる記憶領域として解放する処理とを実行させるプログラムを記憶する記憶手段と、この記憶手段からこのプログラムを読み出してこのプログラムを送信する送信手段とを備えたことを特徴としている。

このような構成とすれば、このプログラムをダウンロードして実行する全てのコンピュータにおいて、ガーベジ・コレクションを、所定の実行主体においてローカルに実行できるため、他の実行主体による通常の処理を停止することなく、

並行して行うことが可能となる。

【 0 0 2 3 】

【 発 明 の 実 施 の 形 態 】

以下、添付図面に示す実施の形態に基づいてこの発明を詳細に説明する。

図 1 は、本実施の形態におけるメモリ管理を実現するコンピュータシステムの全体構成を説明する図である。本実施の形態のコンピュータシステムは、プログラムを複数の実行主体であるスレッドに分けて実行すると共に、複数の当該スレッドがメモリ装置に格納されているオブジェクトを共有してそれぞれアクセスするようなデータ処理環境を有する。以下の説明では、特に、そのようなデータ処理環境として、J a v a 言語で記述されたプログラムの動作環境（J a v a の動作環境）を有するコンピュータシステムを例として説明する。

【 0 0 2 4 】

図 1 において、符号 1 1 0 はオブジェクトプールであり、メモリ装置上においてオブジェクトを格納する記憶領域である。符号 1 2 1 ～ 1 2 4 はオブジェクトである。符号 1 3 1 ～ 1 3 3 はスレッドである。符号 1 4 1 はスレッド 1 3 1 からオブジェクト 1 2 1 への参照ポインタ、符号 1 4 2 はスレッド 1 3 2 からオブジェクト 1 2 2 への参照ポインタ、符号 1 4 3 はスレッド 1 3 3 からオブジェクト 1 2 3 への参照ポインタである。また、符号 1 4 4 はオブジェクト 1 2 1 からオブジェクト 1 2 2 への参照ポインタ、符号 1 4 5 はオブジェクト 1 2 1 からオブジェクト 1 2 4 への参照ポインタである。符号 1 5 1 ～ 1 5 4 はスレッド局所フラグである。

【 0 0 2 5 】

通常の動作において、スレッド 1 3 1 ～ 1 3 3 は、各々の処理の必要に応じて、オブジェクトプール 1 1 0 にオブジェクトを生成し、使用する。図 1 を参照すると、スレッド 1 3 1 はオブジェクト 1 2 1 を参照すると共に、オブジェクト 1 2 1 を介してオブジェクト 1 2 2 及びオブジェクト 1 2 4 を参照する。同様に、スレッド 1 3 2 はオブジェクト 1 2 2 を参照し、スレッド 1 3 3 はオブジェクト 1 2 3 を参照する。

この状態をオブジェクト側から見ると、オブジェクト 1 2 1 はスレッド 1 3 1



のみから直接参照される。またオブジェクト122は、スレッド132から直接参照されると共に、オブジェクト121を介してスレッド131からも参照される。オブジェクト123はスレッド133のみから直接参照される。オブジェクト124は、オブジェクト121を介してスレッド131のみから参照される。

## 【0026】

ここで、オブジェクトが単一のスレッドのみから直接参照されている状態を、以下の説明では、スレッド局所性を有すると呼ぶ。

すなわち、図1の例では、スレッド131のみから直接参照されているオブジェクト121と、スレッド133のみから直接参照されているオブジェクト123とがスレッド局所性を有する。なお、オブジェクト124は、スレッド131のみから参照されているのであるが、オブジェクト121を介して参照されているので、スレッド局所性を有さない。オブジェクト124のように、他のオブジェクトから参照されているオブジェクトを、スレッド局所性を有さないこととするのは、このようなオブジェクトが実際に単一のスレッドのみから参照されているのか、当該他のオブジェクトを介して複数のスレッドから参照されているのかを判別する作業が繁雑となるからである。

## 【0027】

各オブジェクト121、122、123、124は、自オブジェクトがスレッド局所性を有するかどうかを示すスレッド局所フラグ151、152、153、154を持つ。本実施の形態では、オブジェクト121、122、123、124がスレッド局所性を有する場合にスレッド局所性フラグ151、152、153、154を立て（オン）、スレッド局所性を有さない場合にスレッド局所性フラグ151、152、153、154を倒す（オフ）こととする。

## 【0028】

スレッド131、132、133は、オブジェクトを生成する際に、当該オブジェクトのスレッド局所フラグをオンにセットする。生成されたオブジェクトは、当該オブジェクトを生成したスレッド131、132、133によって他のスレッドまたは他のオブジェクトから参照可能な状態に変化させられる際に、その状態変化に先立って自オブジェクトのスレッド局所フラグをオフにする。スレ

ド131、132、133が、自スレッドに対してスレッド局所性を有するオブジェクトを、他のスレッドまたは他のオブジェクトから参照可能な状態に変化させるべき状況が発生したことを検出する方法としては種々の方法が考えられる。例えば、当該オブジェクトが他のオブジェクトもしくはグローバルなデータに代入された場合に、状態を変化させることとすることができる。さらにこのとき、代入先のオブジェクトが局所的かどうかまで総括的にチェックして解除するようにしても良い。

なお、一度オフにしたスレッド局所フラグを再びオンにすることはない。実際には、所定のスレッドに対してスレッド局所性を有していたオブジェクトが、一時的に他のスレッドから参照された後、当該参照ポインタが消去されて再びスレッド局所性を回復する場合がある。しかし、このような場合分けを行うと、スレッド局所性を判断するための処理が複雑となるので、スレッド局所フラグはオフにしたままとする。

#### 【0029】

図2は、スレッド131、132、133がオブジェクト121、122、123を生成する動作を説明するフローチャートである。

図2を参照すると、スレッド131、132、133は、まず、オブジェクトプール110の所定の領域をオブジェクト領域として確保し、初期化する（ステップ201）。次に、当該オブジェクト領域にスレッド局所フラグ151、152、153を設定し、オンにセットする（ステップ202）。そして、生成されたオブジェクト121、122、123への参照ポインタをスレッド131、132、133内に格納する（ステップ203）。

#### 【0030】

このようにして、オブジェクト121、122、123は、生成された時点では全てスレッド局所フラグ151、152、153がオンとなっている。図1は、この後、オブジェクト122がスレッド局所性を失い、スレッド局所フラグ152がオフとなった状態を示している。なお、図1の例では、オブジェクト122はオブジェクト121から参照可能となっている。

また、図1に示すオブジェクト124は、いずれかのスレッド131、132

、133により生成された後、スレッド局所性を失ってスレッド局所フラグ154がオフとなり、さらにその後、オブジェクト121から参照可能な状態となり、他の参照ポインタが消去されてオブジェクト121からの参照のみが残った状態である。

#### 【0031】

Javaのように複数のスレッドがオブジェクトを共有してそれぞれアクセスするようなデータ処理環境では、所定のスレッドが所定のオブジェクトを参照している間は、他のスレッドが当該オブジェクトを参照できないようにする必要がある。そこで、各スレッドは、オブジェクトを参照する際にロック（同期）処理を行って、他のスレッドが当該オブジェクトにアクセスすることを拒絶するようにする。他のスレッドのロック（同期）処理によってオブジェクトへのアクセスを拒絶されたスレッドは、当該オブジェクトがアンロック（解放）されるまで当該オブジェクトの参照を待ち合わせる事となる。

#### 【0032】

しかしながら、本実施の形態におけるスレッド局所性を有するオブジェクトは、当該オブジェクトを参照する主体が単一のスレッドに限定されている。したがって、スレッドが自スレッドに対してスレッド局所性を有するオブジェクトを参照する場合に関しては、ロック（同期）処理を省略することができる。

#### 【0033】

図3は、スレッド131、132、133がオブジェクト121、122、123、124を参照する際のロック（同期）処理に関わる動作を説明するフローチャートである。

図3を参照すると、スレッド131、132、133は、まず、参照しようとするオブジェクト121、122、123、124のスレッド局所フラグ151、152、153、154がオンにセットされているかどうかを調べる（ステップ301）。そして、スレッド局所フラグ151、152、153、154がオンであれば、その後の処理を省略して当該オブジェクト121、122、123、124を参照する。ただし、このとき、必要に応じてロック（同期）処理の省略に伴う処理を実行する（ステップ305）。スレッド131がオブジェクト1

21を参照する場合、及びスレッド133がオブジェクト123を参照する場合がこれに該当する。ロック（同期）処理の省略に伴う処理の内容については後述する。

#### 【0034】

これに対し、スレッド局所フラグ151、152、153、154がオフである場合は、通常通り、ロック（同期）処理を行う。すなわち、オブジェクト121、122、123、124を参照しようとするスレッド131、132、133は、まず、当該オブジェクト121、122、123、124が他のスレッド131、132、133によりロックされているかどうかを調べる（ステップ302）。そして、ロックされている場合は、アンロックされるまで待つ（ステップ304）。一方、当該オブジェクト121、122、123、124が他のスレッド131、132、133によりロックされていない場合（アンロックされた場合を含む）は、当該オブジェクト121、122、123、124に対してロック（同期）処理を行い（ステップ303）、参照する。スレッド132がオブジェクト122を参照する場合、及びスレッド131がオブジェクト121を介してオブジェクト122またはオブジェクト124を参照する場合がこれに該当する。

#### 【0035】

本実施の形態において、スレッドがオブジェクトを参照する際のロック（同期）処理を省略した場合は、当該オブジェクトを参照した後に行うアンロック（解放）処理も省略できる。

図4は、スレッド131、132、133がオブジェクト121、122、123、124を参照した後のアンロック（解放）処理に関わる動作を説明するフローチャートである。

図4を参照すると、スレッド131、132、133は、まず、参照したオブジェクト121、122、123、124のスレッド局所フラグ151、152、153、154がオンにセットされているかどうかを調べる（ステップ401）。そして、スレッド局所フラグ151、152、153、154がオンであれば、その後の処理を省略して当該オブジェクト121、122、123、124

の参照を終了する。ただし、このとき、必要に応じてアンロック（解放）処理の省略に伴う処理を実行する（ステップ403）。ロック（同期）処理の省略の場合と同様に、スレッド131がオブジェクト121を参照した場合、及びスレッド133がオブジェクト123を参照した場合がこれに該当する。

一方、スレッド局所フラグ151、152、153、154がオフである場合は、通常通り、アンロック（解放）処理を行う（ステップ402）。

#### 【0036】

次に、オブジェクト121、123のように、スレッド局所フラグがオンにセットされており、特定のスレッドに対してスレッド局所性を有するオブジェクトが、当該スレッド局所性の対象ではない他のスレッドまたはオブジェクトに参照可能な状態に変更される際の動作を説明する。図5は、この際の動作を説明するフローチャートである。

#### 【0037】

スレッド131、132、133が、自スレッドに対してスレッド局所性を有するオブジェクト121、122、123、124に関して、他のスレッドまたは他のオブジェクトから参照可能な状態に変化させるべき状況が発生したことを検出したものとする。上述したように、このような状況を検出する方法としては種々の方法を採用することができるが、ここでは、当該オブジェクトが他のオブジェクトもしくはグローバルなデータに代入された場合に、状態を変化させるべき状況が発生したと認識することとする。

#### 【0038】

この場合、まず、スレッド131、132、133は、まず、アクセスしようとするオブジェクト121、122、123、124のスレッド局所フラグ151、152、153、154がオンにセットされているかどうかを調べる（ステップ501）。そして、スレッド局所フラグ151、152、153、154がオフであれば、何ら特別の処理を行うことなく、オブジェクト121、122、123、124への参照ポインタを当該オブジェクトを代入しようとした他のオブジェクトやデータに代入する（ステップ504）。オブジェクト122、124が他のオブジェクトまたはグローバルなデータに代入された場合がこれに該当

する。

【0039】

これに対し、スレッド局所フラグ151、152、153、154がオンである場合、スレッド131、132、133は、まず、当該オブジェクト121、122、123、124に対して、当該スレッド局所フラグ151、152、153、154をオフにするように指示する。これに応じて、オブジェクト121、122、123、124が当該スレッド局所フラグ151、152、153、154をオフにすると（ステップ502）、当該オブジェクト121、122、123、124のスレッド局所性が失われることとなるが、これに伴って、以下の処理を行うことが必要となる。

【0040】

例として、スレッド133に対してスレッド局所性を有するオブジェクト123がスレッド局所性を解除される場合について考える。すなわち、オブジェクト123のスレッド局所性が解除された後、他のスレッドまたは他のオブジェクトがオブジェクト123をロックしようとする場合、当該オブジェクト123がスレッド局所性の対象であったスレッド133によりロックされていたならば、スレッド132はオブジェクト123がアンロック（解放）されるまで参照を待たなければならない。しかし、スレッド132が参照しようとするまで、オブジェクト123はスレッド133に対してスレッド局所性を有していたため、スレッド133による参照の際にはロック（同期）処理及びアンロック（解放）処理が省略されている。したがって、スレッド132による参照に先立って、省略されていたロック（同期）処理及びアンロック（解放）処理を実行して、処理における整合性を取る必要がある。

【0041】

したがって、スレッド局所フラグ151、152、153、154をオフにした後、当該オブジェクト121、122、123、124におけるスレッド局所性の対象であったスレッド131、132、133は、省略されていた当該オブジェクト121、122、123、124におけるロック（同期）処理及びアンロック（解放）処理を実行する（ステップ503）。そして、整合性がとれた後

に、オブジェクト 1 2 1、1 2 2、1 2 3、1 2 4 への参照ポインタを当該オブジェクトを代入しようとした他のオブジェクトやデータに代入する（ステップ 5 0 4）。

なお、この整合性確保のための処理は、オブジェクト 1 2 1、1 2 2、1 2 3、1 2 4 がスレッド局所性を失う直前に、当該オブジェクト 1 2 1、1 2 2、1 2 3、1 2 4 におけるスレッド局所性の対象であったスレッド 1 3 1、1 3 2、1 3 3 により実行されるため、Racing等の問題は発生しない。

#### 【 0 0 4 2 】

ステップ 5 0 3 で省略されていたロック（同期）処理及びアンロック（解放）処理を実行する場合、新規にアクセスしようとする要求が発生した時点で省略されているロック（同期）処理及びアンロック（解放）処理を認識することが必要となる。この認識を行う手法は任意であるが、ここでは次の二つの方法を例示する。

#### 【 0 0 4 3 】

第 1 の手法は、スレッド局所性の対象であったスレッドの実行スタックをスキャンして、省略されているロック（同期）処理及びアンロック（解放）処理を認識する方法である。この場合、オブジェクトが参照されるたびに、スレッドの実行スタックには参照履歴が自動的に格納されるので、図 3 のフローチャートにおけるロック（同期）処理の省略に伴う処理（ステップ 3 0 5）及び図 4 のフローチャートにおけるアンロック（解放）処理の省略に伴う処理（ステップ 4 0 3）は実体的な処理が存在しないこととなる。しかし、オブジェクトにおけるスレッド局所性が失われる際に、スレッドの実行スタックをスキャンすることによるオーバーヘッドが生じる。

#### 【 0 0 4 4 】

第 2 の手法は、ロック（同期）処理及びアンロック（解放）処理が省略された際に、省略された回数をカウントしておき、オブジェクトのスレッド局所性が失われる際に、当該カウント値に基づいて省略された処理を実行する方法である。この場合、図 3 のフローチャートにおけるロック（同期）処理の省略に伴う処理（ステップ 3 0 5）及び図 4 のフローチャートにおけるアンロック（解放）処理

の省略に伴う処理（ステップ403）の内容が、省略された回数をカウントする処理となる。この手法によれば、オブジェクトにおけるスレッド局所性が失われる際に、スレッドの実行スタックをスキャンすることによるオーバーヘッドは生じない。また、ステップ305、403で実行されるカウント処理は、省略されたロック（同期）処理及びアンロック（解放）処理よりもはるかに軽いため、システムに対する負担を大幅に軽減することができる。

## 【0045】

上述したように、複数のスレッドがオブジェクトを共有してそれぞれアクセスするようなデータ処理環境では、オブジェクトプールに、いずれのスレッドからも参照されないオブジェクトが生じる場合がある。すなわち、スレッドは、種々の処理を実行するためにオブジェクトを生成するが、その後、当該オブジェクトを参照する必要が無くなり、参照ポインタが消去されると、当該オブジェクトは全く処理の実行に関与しない状態でオブジェクトプールに放置される。したがって、このような放置されたオブジェクトを効率的に回収し、空き領域として使用するガーベジ・コレクションを行うことが重要となる。

## 【0046】

このような放置されたオブジェクトを回収するには、オブジェクトがいずれのスレッドからも参照されていないことを確認しなければならない。したがって、一般に、全てのスレッドの参照ポインタ群をたどって、参照の有無を調べることが必要となる。

しかし、本実施の形態は、特定のスレッドに対してスレッド局所性を有するかどうかを示すスレッド局所フラグをオブジェクトに付加しているため、スレッド局所フラグがオンにセットされているオブジェクトでは、当該スレッド局所性の対象であるスレッドだけが当該オブジェクトを参照することが保証されている。したがって、当該スレッド内に当該オブジェクトへの参照ポインタが残っているかどうかを調べるだけで、当該オブジェクトを回収できるかどうかを判断することができる。

## 【0047】

図6は、スレッド局所フラグを用いたガーベジ・コレクションの手順の一例を



説明するフローチャートである。

図 6 を参照すると、ガーベジ・コレクションを実行しようとするスレッドは、自スレッドに対してスレッド局所性を有するオブジェクトを一つ選択する（ステップ 6 0 1）。そして、当該オブジェクトへの参照ポインタが自スレッド内にあるかどうかを調べ（ステップ 6 0 3）、参照ポインタがない場合には当該オブジェクトを回収する（ステップ 6 0 4）。

以上の動作を繰り返して行い、自スレッドに対して局所性を有する全てのオブジェクトについて調べたならば、処理を終了する（ステップ 6 0 2）。

#### 【 0 0 4 8 】

以上のようにして、所定のスレッドとの関係においてスレッド局所フラグをオンにセットしていたオブジェクトのうちで、当該スレッドから参照されない（すなわち、いずれのスレッドからも参照されない）オブジェクトを回収することができる。この処理は、スレッド自身のスタックを 1 段スキャンするだけで所定のオブジェクトを回収できるかどうかを判断でき、オブジェクト内部の参照ポインタのトレースを必要としないため、極めて高速に実行することができる。また、この処理は、所定のスレッドがローカルに実行するため、他のスレッドによる通常の処理を停止させることなく、並行して実行することができる。

#### 【 0 0 4 9 】

なお、本実施の形態によるスレッド局所フラグを用いたガーベジ・コレクションを、各スレッドにおいて順次実行したとしても、スレッド局所フラグがオフであっていずれのスレッドからも参照されないようなオブジェクトは回収することができない。そのようなオブジェクトは、全てのスレッドによる処理を一度停止させ、オブジェクトプール中の全てのオブジェクトの中からいずれのスレッドからも参照されないオブジェクトを検索するといった、従来の方法により、ガーベジ・コレクションを実行すれば良い。すなわち、本実施の形態によるスレッド局所フラグを用いたガーベジ・コレクションは、従来技術によるガーベジ・コレクションと併用することが可能である。

#### 【 0 0 5 0 】

#### 【発明の効果】

以上説明したように、本発明によれば、単一の実行主体からのみアクセスが行われ、他の実行主体からアクセス要求される可能性のないデータに対しては、アクセスする際にロック（同期）処理を省略することにより、システムの負荷を軽減し、全体的な性能を向上させることができる。

【 0 0 5 1 】

また、通常の処理と並行して行うガーベジ・コレクションの効率化を図ることにより、システムの全体性能の低下を軽減することができる。

【図面の簡単な説明】

【図 1】 本実施の形態におけるメモリ管理を実現するコンピュータシステムの全体構成を説明する図である。

【図 2】 本実施の形態におけるスレッドがオブジェクトを生成する動作を説明するフローチャートである。

【図 3】 本実施の形態におけるスレッドがオブジェクトを参照する際のロック（同期）処理に関わる動作を説明するフローチャートである。

【図 4】 本実施の形態におけるスレッドがオブジェクトを参照した後のアンロック（解放）処理に関わる動作を説明するフローチャートである。

【図 5】 本実施の形態における特定のスレッドに対してスレッド局所性を有するオブジェクトがスレッド局所性を解除される際の動作を説明するフローチャートである。

【図 6】 本実施の形態におけるスレッド局所フラグを用いたガーベジ・コレクションの手順の一例を説明するフローチャートである。

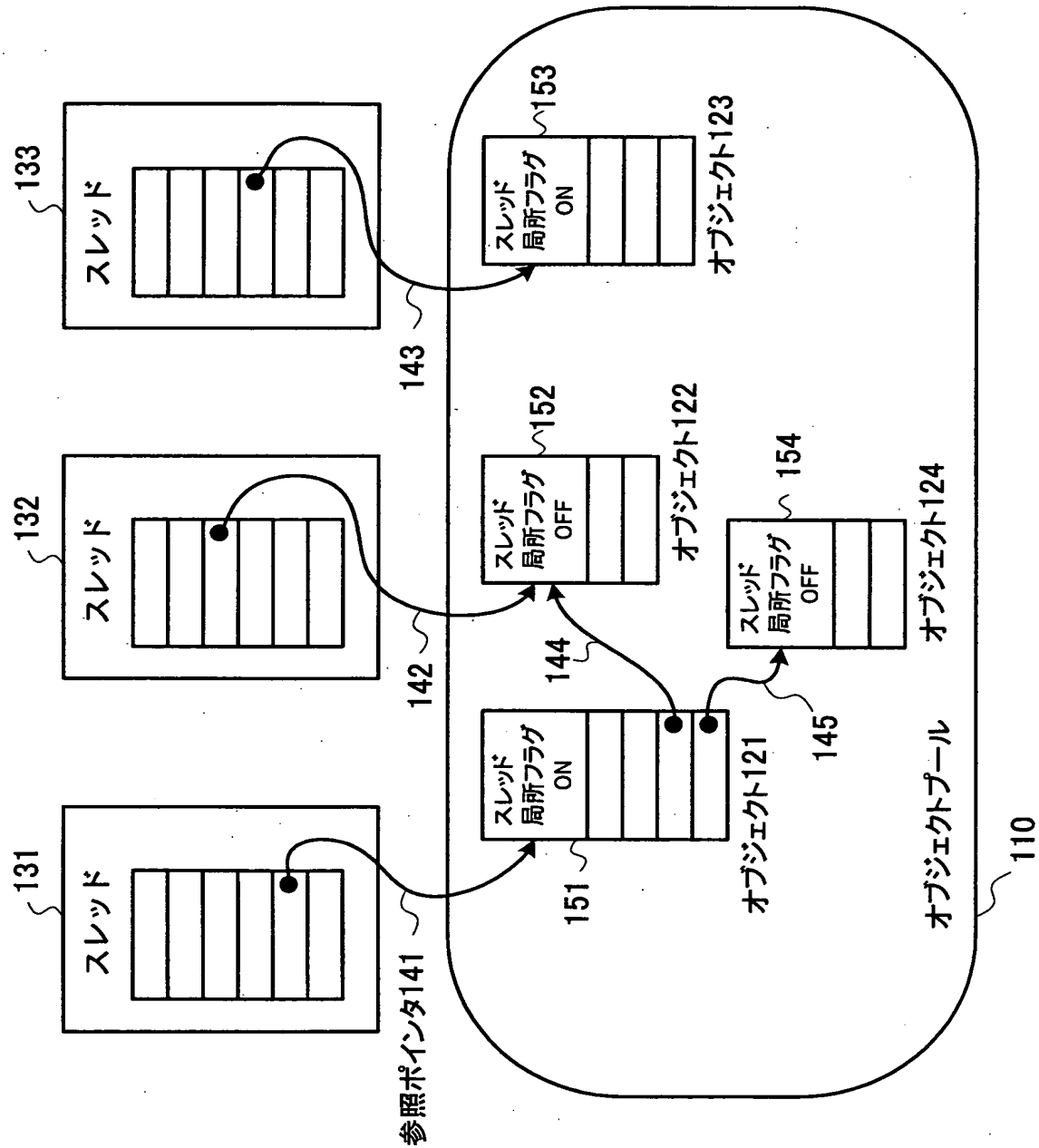
【符号の説明】

1 1 0 …オブジェクトプール、1 2 1、1 2 2、1 2 3、1 2 4 …オブジェクト、1 3 1、1 3 2、1 3 3 …スレッド、1 4 1、1 4 2、1 4 3、1 4 4、1 4 5 …参照ポインタ、1 5 1、1 5 2、1 5 3、1 5 4 …スレッド局所フラグ

【書類名】

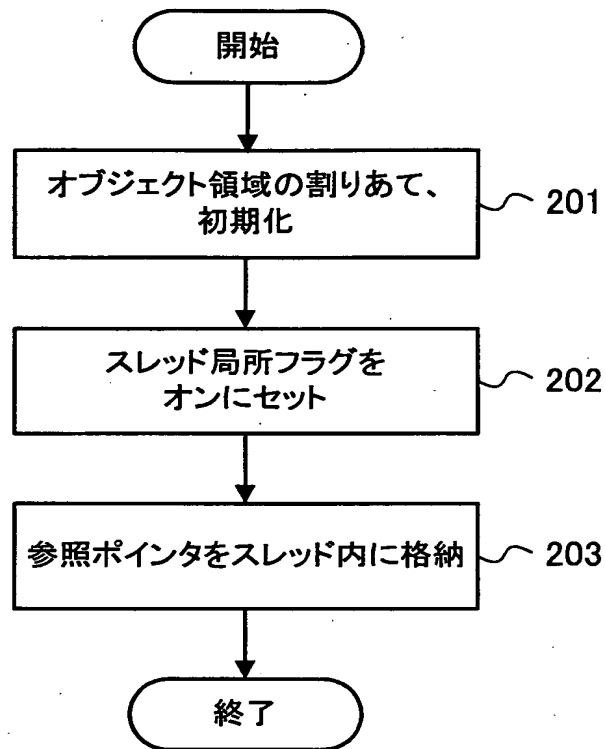
図面

【図 1】



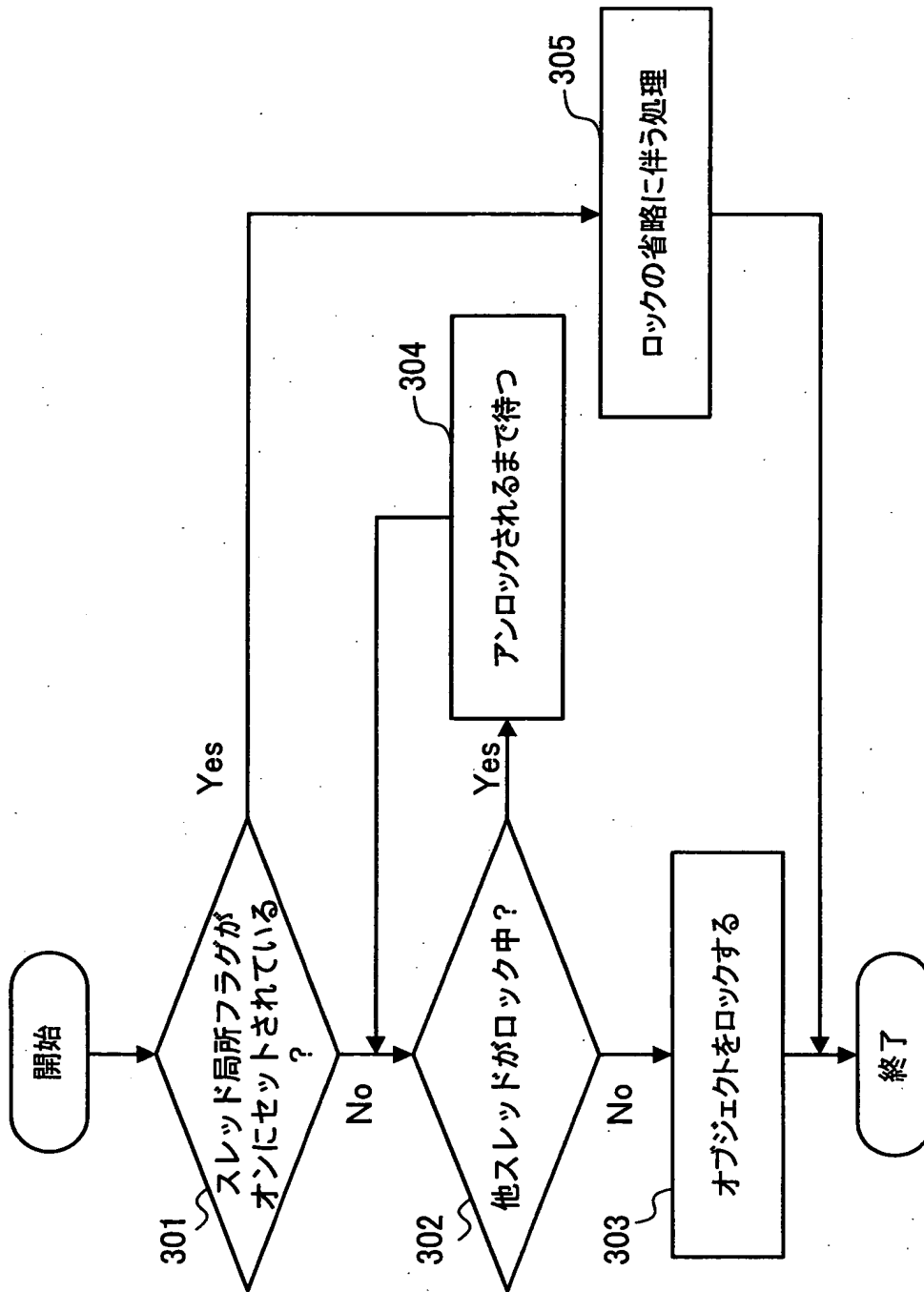
【図 2】

オブジェクトの生成処理



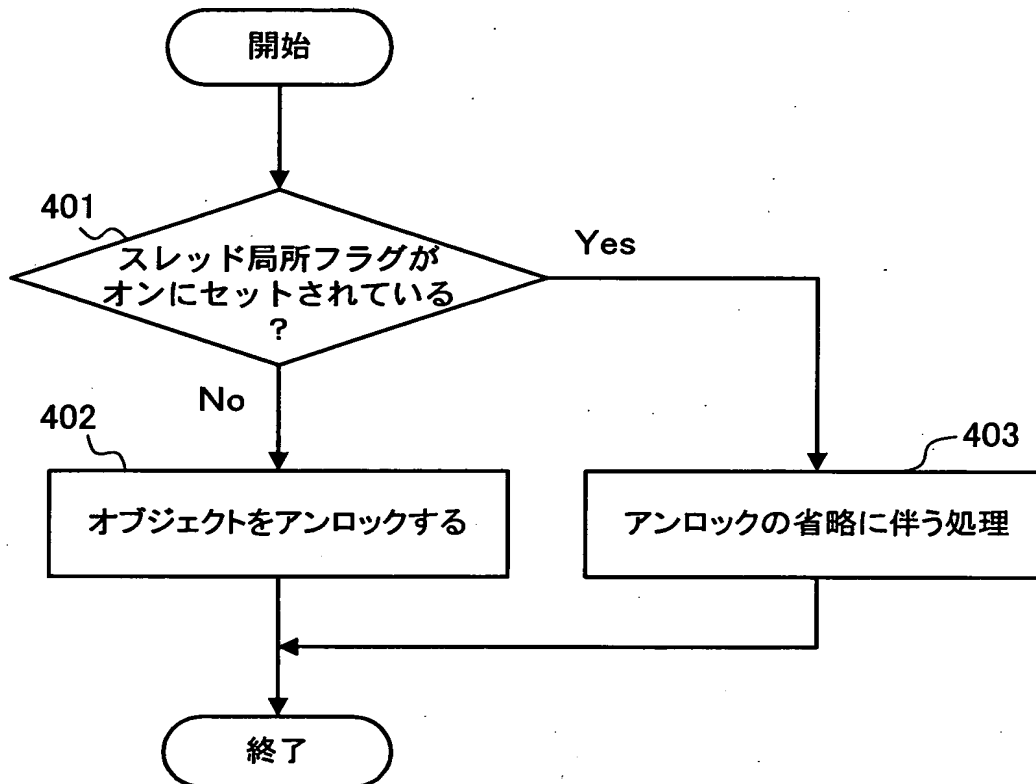
【図 3】

オブジェクトのロックに関わる処理



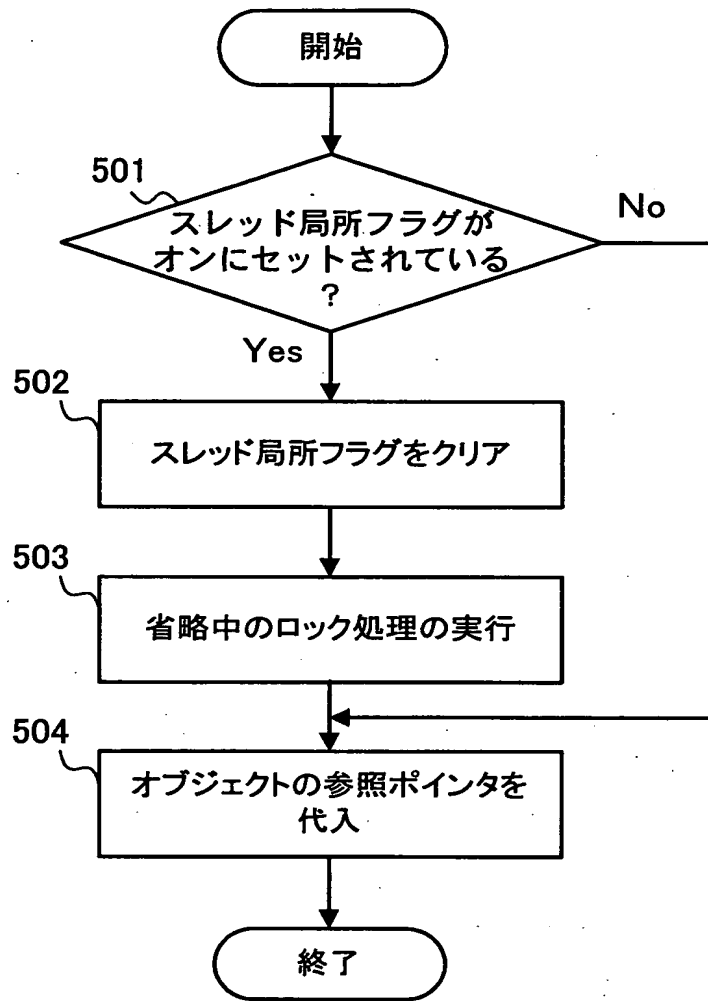
【図4】

オブジェクトのアンロックに関わる処理



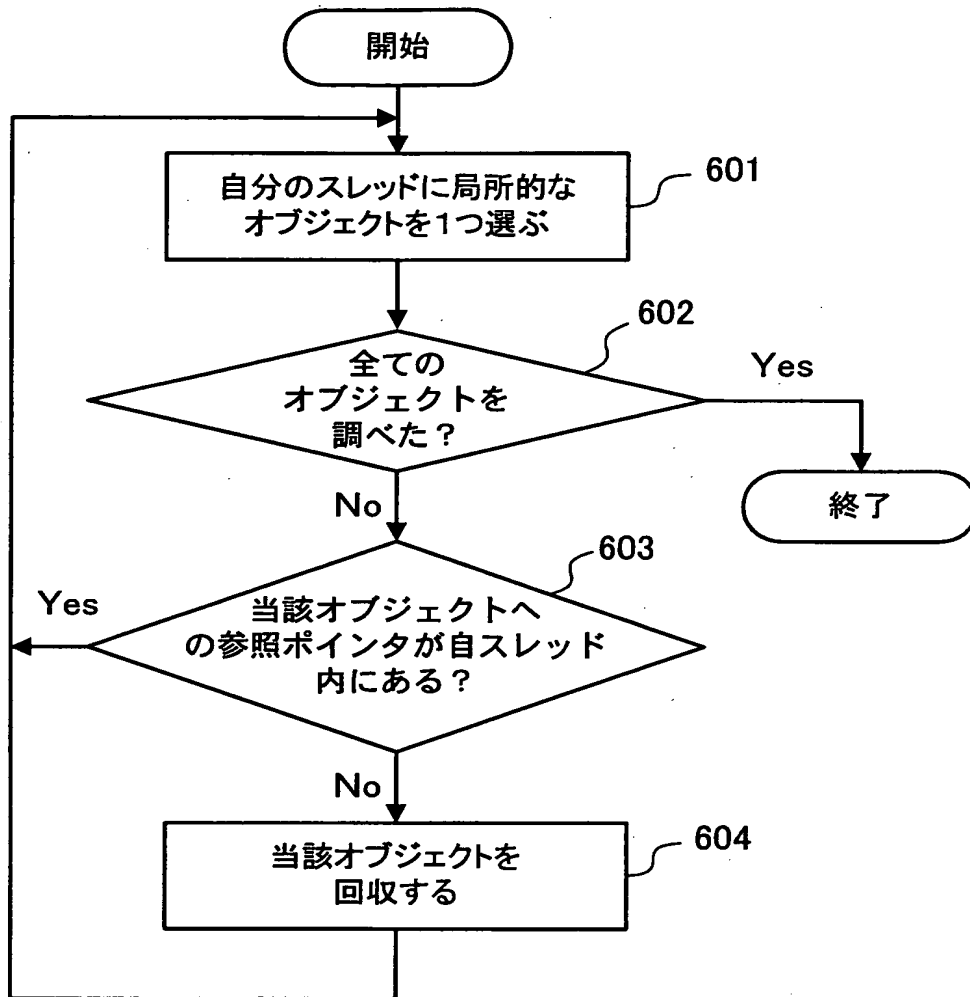
【図 5】

オブジェクトがスレッド局所性を  
解除される際の動作



【図 6】

スレッド局所フラグを用いた  
ガーベジ・コレクション





【書類名】 要約書

【要約】

【課題】 スレッドがオブジェクトに対してアクセスする際に、ロック（同期）処理を省略することにより、システムの負荷を軽減し全体的な性能を向上させる。

【解決手段】 複数のスレッド 1 3 1、1 3 2、1 3 3 がオブジェクト 1 2 1、1 2 2、1 2 3、1 2 4 を共有してそれぞれアクセスするようなデータ処理環境を有するコンピュータシステムにおいて、オブジェクト 1 2 1、1 2 2、1 2 3、1 2 4 はスレッド局所性の有無を示すスレッド局所フラグ 1 5 1、1 5 2、1 5 3、1 5 4 を有し、スレッド 1 3 1、1 3 2、1 3 3 は、アクセスしようとするオブジェクト 1 2 1、1 2 2、1 2 3、1 2 4 におけるスレッド局所フラグ 1 5 1、1 5 2、1 5 3、1 5 4 が、このスレッド 1 3 1、1 3 2、1 3 3 に対して局所性を有することを示すか否かに応じて、ロック（同期）処理を行わずにオブジェクト 1 2 1、1 2 2、1 2 3、1 2 4 にアクセスする。

【選択図】 図 1

認定・付加情報

特許出願の番号	特願 2000-064718
受付番号	50000279561
書類名	特許願
担当官	濱谷 よし子 1614
作成日	平成 12 年 4 月 20 日

<認定情報・付加情報>

【特許出願人】

【識別番号】	390009531
【住所又は居所】	アメリカ合衆国 10504、ニューヨーク州 アーモンク (番地なし)
【氏名又は名称】	インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】	100086243
【住所又は居所】	神奈川県大和市下鶴間 1623 番地 14 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	坂口 博

【復代理人】

【識別番号】	申請人
【識別番号】	100104880
【住所又は居所】	東京都港区赤坂 7-10-9 第 4 文成ビル 202 セリオ国際特許事務所
【氏名又は名称】	古部 次郎

【選任した代理人】

【識別番号】	100091568
【住所又は居所】	神奈川県大和市下鶴間 1623 番地 14 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	市位 嘉宏

【選任した復代理人】

【識別番号】	100100077
【住所又は居所】	東京都港区赤坂 7-10-9 第 4 文成ビル 202 セリオ国際特許事務所
【氏名又は名称】	大場 充

出 願 人 履 歴 情 報

識別番号 [390009531]

1. 変更年月日 1990年10月24日  
[変更理由] 新規登録  
住 所 アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)  
氏 名 インターナショナル・ビジネス・マシーンズ・コーポレーション
2. 変更年月日 2000年 5月16日  
[変更理由] 名称変更  
住 所 アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)  
氏 名 インターナショナル・ビジネス・マシーンズ・コーポレーション